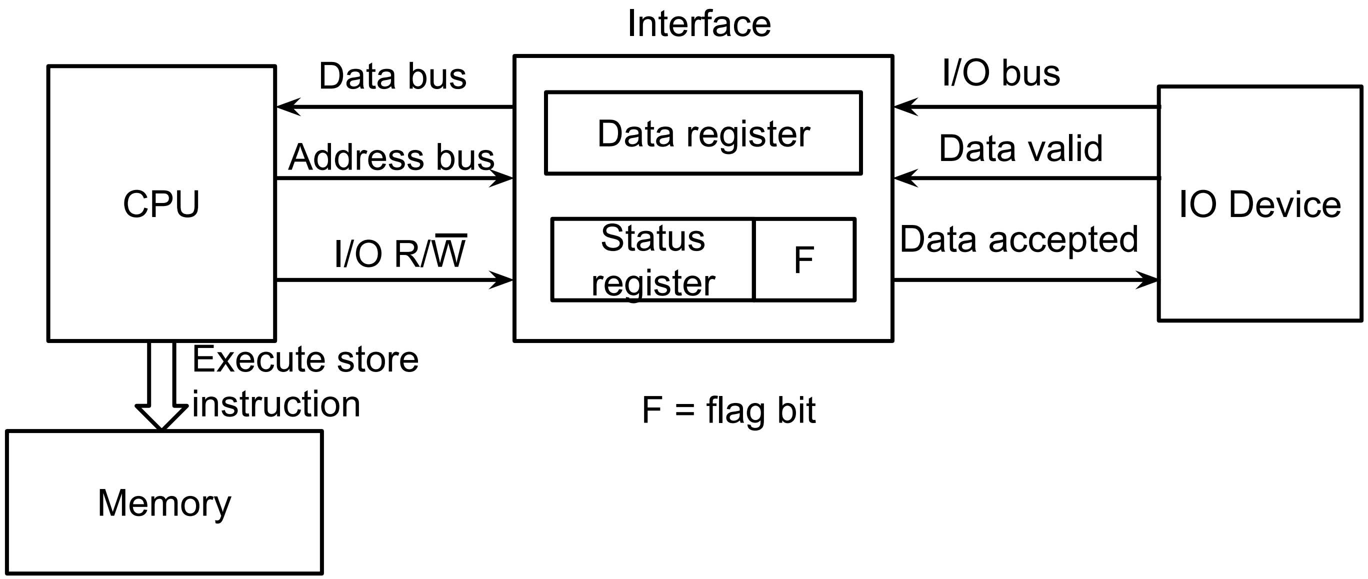
Modes of data transfer

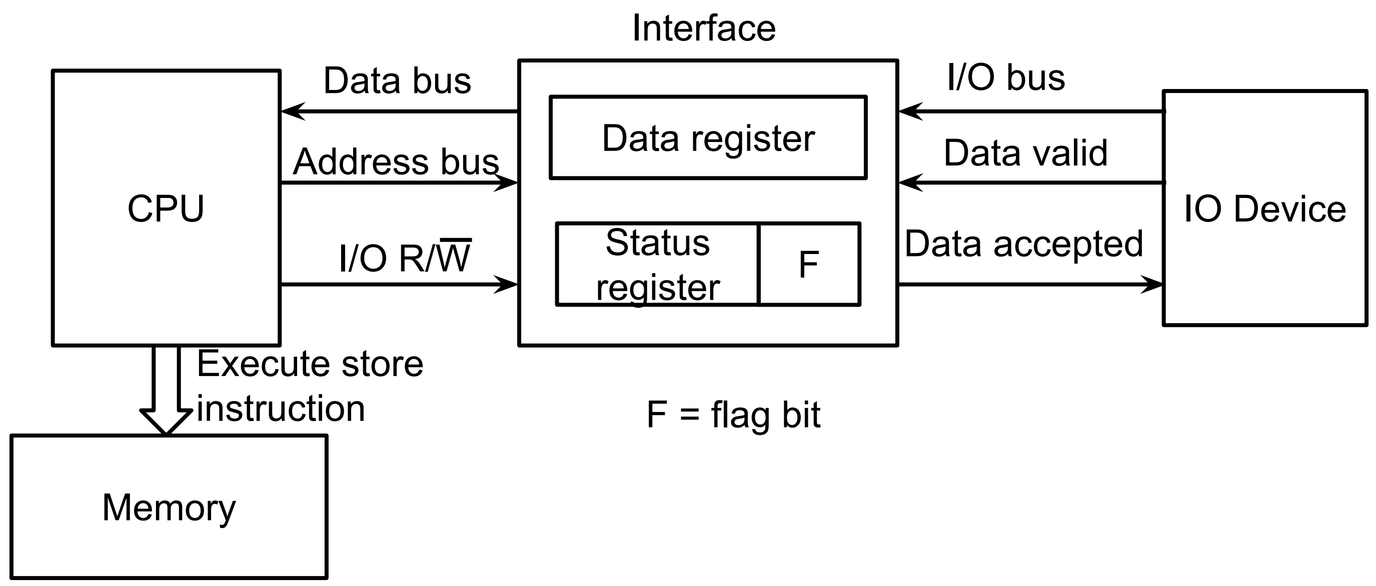
1. Programmed I/O
2. Interrupt initiated I/O
3. Direct Memory Access (DMA)

Programmed I/O



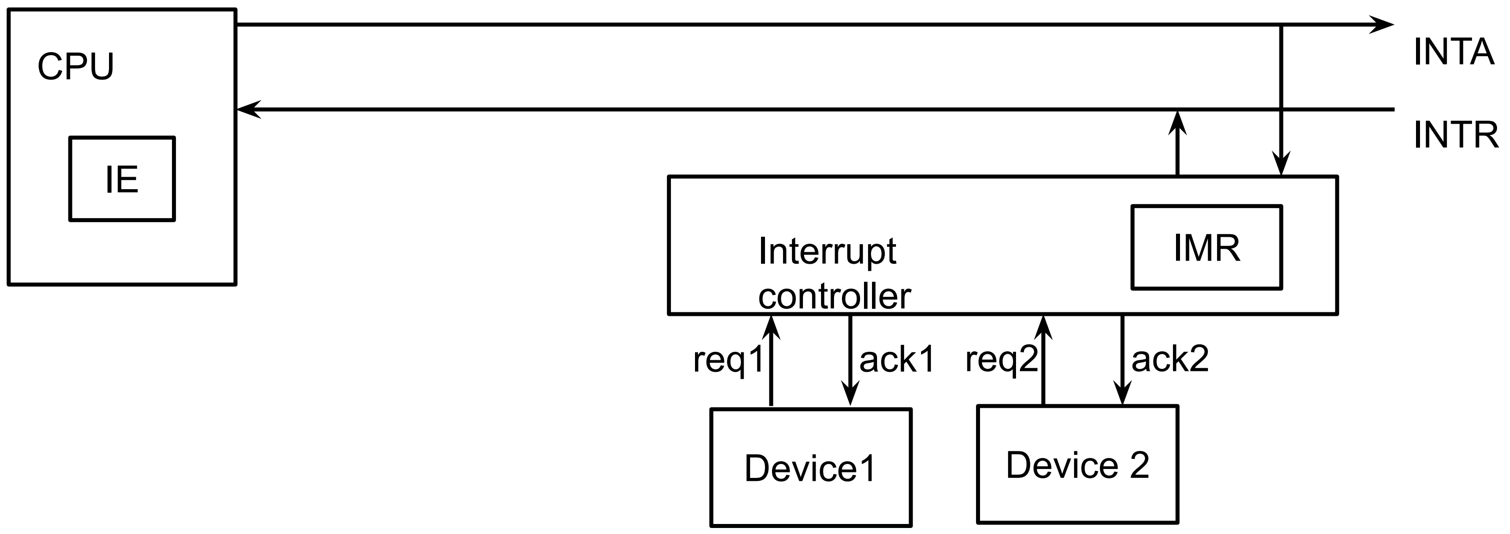
- This is a software method where data are transferred between memory and I/O device with the presence of CPU.
- This is the example of transferring data from I/O device to memory.
- A transfer from an I/O device to memory requires the write instruction executed by CPU.
- The interface is used in between I/O device and CPU.
- The handshaking procedure is followed here.
- The I/O device place the data byte in I/O bus and enables the data valid line.
- The interface accepts the data byte and place into Data register. There after it enables the Data accepted line. The flag bit is then set.
- The I/O device disables the Data valid line and not sending the next data byte until Data accepted line will be disabled.

Programmed I/O contd.



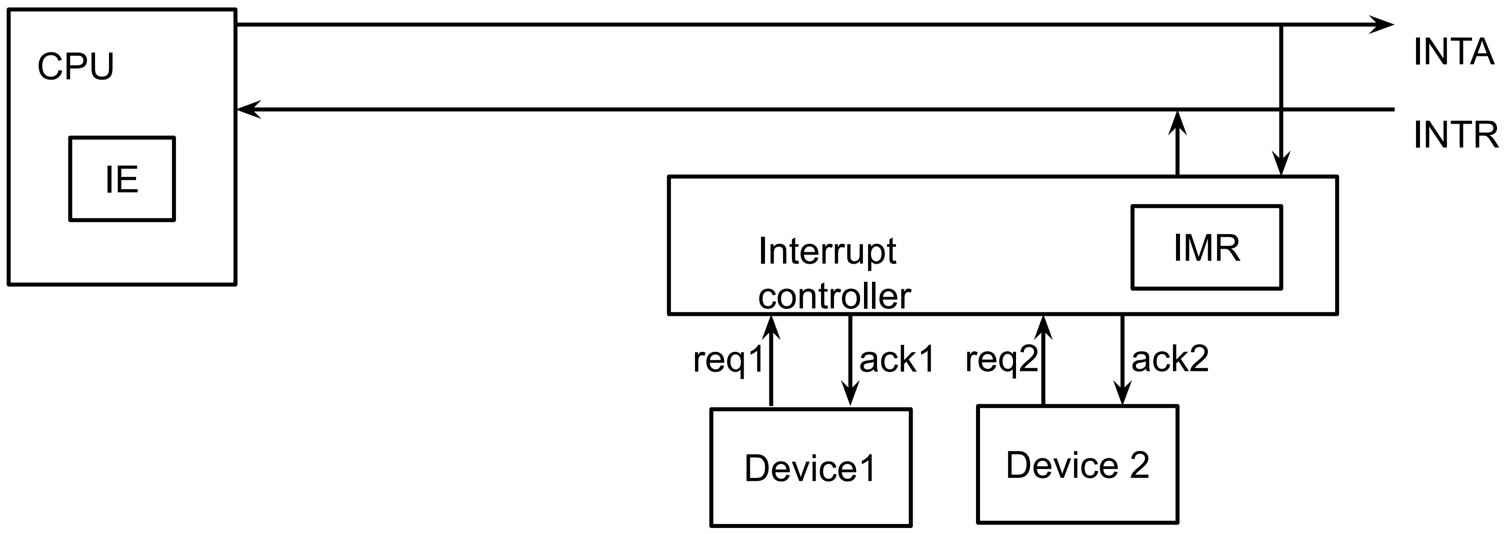
- A program is used to continuously checking the flag bit of the status register and determine the data byte is placed in the Data register or not.
- If the flag bit is 1 then CPU activates I/O read operation and generate the address of data register.
- Then data byte is transferred from Interface to CPU through data bus and flag bit is reset to 0.
- After resetting the fag bit the data accepted line is disabled and the interface is ready to take next data byte.
- The CPU transfer this data byte to memory by executing store instruction.
- Disadvantage of the programmed I/O method is the program continuously checks the flag bit of status register and CPU is unnecessarily busy in executing the program until I/O device transfers the data byte to interface.

Interrupt-initiated I/O



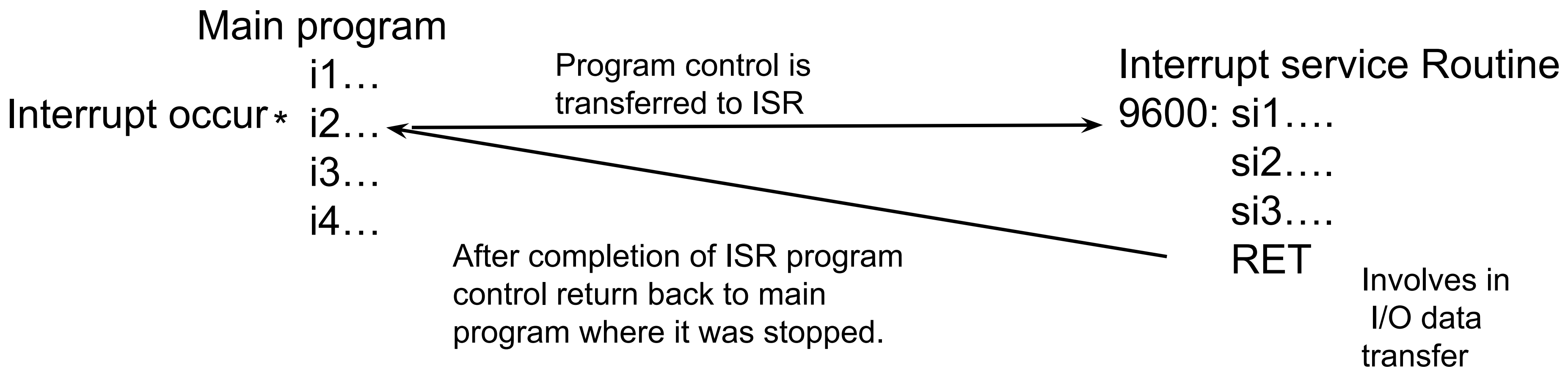
- This is a software method where Interrupt controller is used to handle the data transfer.
- When I/O device is ready to data transfer it send the req (request) signal to interrupt controller.
- After getting the req signal Interrupt controller activates the INTR signal (Interrupt request) to CPU.
- A program is a sequence of instructions. The CPU executes one by one instruction.
- When INTR line is high CPU checks the interrupt enable (IE) bit.
- If IE bit is 1 then CPU stops the execution of current instruction and activate the INTA (Interrupt Acknowledge) signal.
- After getting the INTA signal Interrupt controller activates the ack signal to I/O device.
- There after program control will be transferred to the Interrupt Service Routine (ISR).
- ISR is used for data transfer. After the completion of ISR execution the program control again return back to the main program where it is stopped.

Interrupt-initiated I/O



- The interrupt controller uses a register called Interrupt Mask Register(IMR) to detect any Interrupt from the I/O devices.
- Consider there are n number of IO devices in the system. Therefore IMR is n bit register, where each bit indicates the status of one I/O device.
- Let, the content of IMR is denoted as E_0, E_1….. E_{n-1}. When E_0 =1 then interrupt from device 0 is recognized; When E_1=1 then interrupt from device1 is recognized and so on.

Interrupt Service Routine (ISR)



Main program

```
                   i1…
Interrupt occur *  i2…
                   i3…
                   i4…
```

Program control is transferred to ISR

Interrupt service Routine

```
9600: si1….
      si2….
      si3….
      RET
```

After completion of ISR program control return back to main program where it was stopped.

Involves in I/O data transfer

Here i indicates the instruction and si indicates service instruction.

Types of Interrupt

- Depending on functionality

i) Maskable interrupt : This type of interrupt can be disabled or delayed.
Ex: RST 7.5, RST6.5

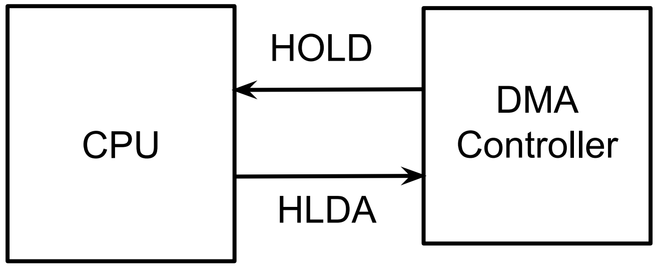
i) Non-maskable interrupt: This type of interrupt can't be ignored or delayed it will obviously occur. Ex: TRAP

- Depending on ISR

i) Vectored interrupt: here starting address of ISR is known prior. EX: RST 7.5

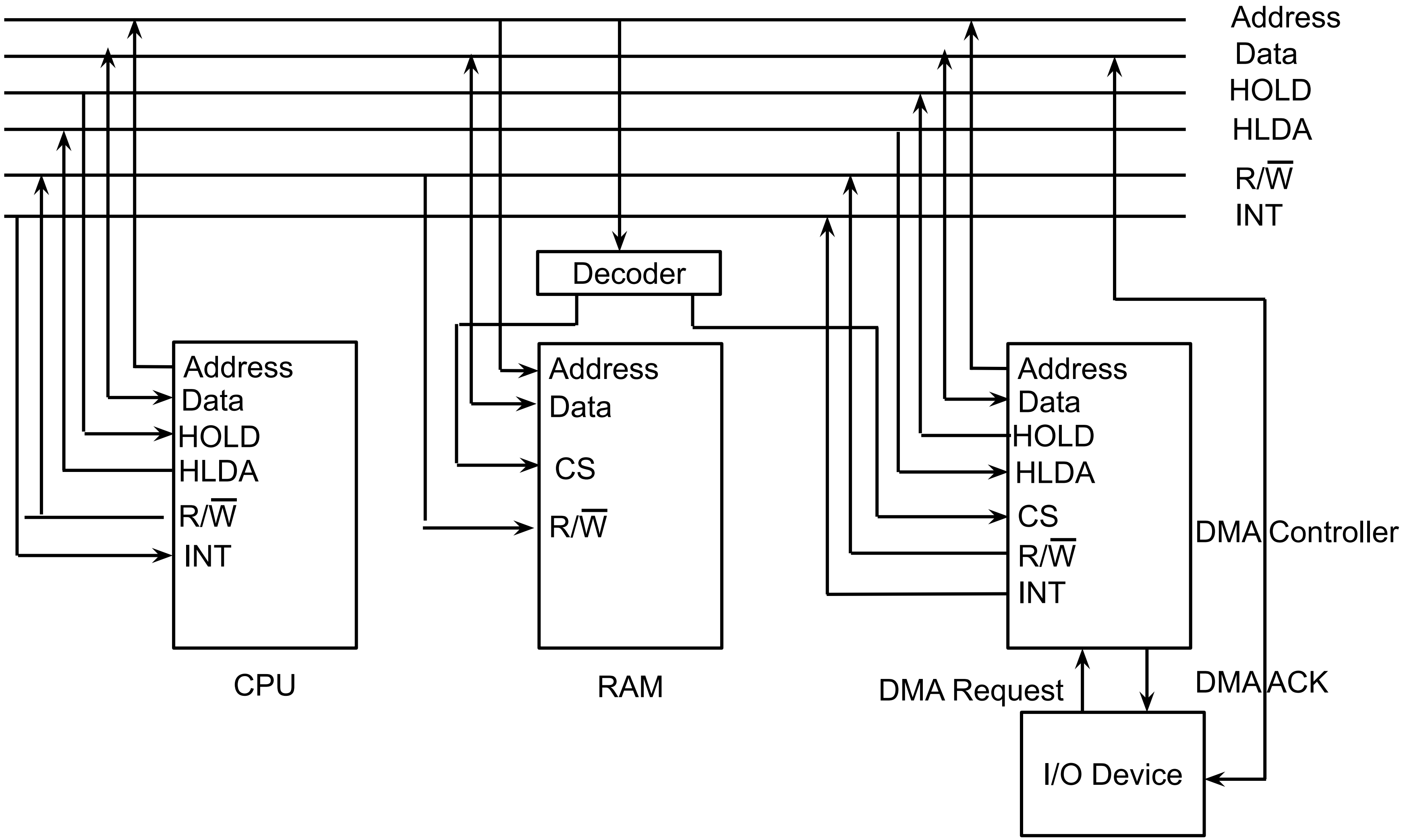
ii) Non-vectored interrupt: here starting address is unknown which will be supplied by external device. EX: INTR

Direct Memory Access



- Data transfer between the main memory and an I/O device (may be hard disk) occurs without the involvement of CPU.
- This is a hardware method.
- To transfer the large block of data at high speed DMA operation is used.
- Beginning of DMA operation DMA controller sends the HOLD signal to CPU
- After getting the HOLD signal CPU release the control over address bus and data bus by changing their state to high impedance state.
- Then CPU generates the hold acknowledgement (HLDA) signal to DMA controller.
- The control of the busses has been taken by the DMA controller.
- Then DMA controller starts other functionalities.

DMA Transfer Operation



DMA transfer operation

- I/O device sends a DMA request signal to DMA controller.
- To transfer the large block of data at high speed DMA operation is used.
- Beginning of DMA operation DMA controller sends the HOLD signal to CPU.
- After getting the HOLD signal CPU release the control over address bus and data bus by changing their state to high impedance state.
- Then CPU generates the hold acknowledgement (HLDA) signal to DMA controller.
- The control of the busses has been taken by the DMA controller.
- Before releasing the control of the buses to the controller, the CPU initializes the address register for starting memory address of the block of data.
- Word count register is initialized to the number of words to be transferred in operation type (read or write).
- The I/O device then communicate with memory through the direct data bus.
- For each word transferred, the DMA controller increment the address register and decrements word count register.
- When the word count register reaches zero, the DMA controller stops any further transfer.
- The termination of DMA operation is informed to the CPU by using INT signal.
- Then CPU again acquires the control over address bus and data bus.

Advantage of DMA

- High speed data transfer is possible, since CPU is not involved during actual transfer, which occurs between I/O device and the main memory.
- Parallel processing can be achieved between CPU processing and DMA controller’s I/O operation.

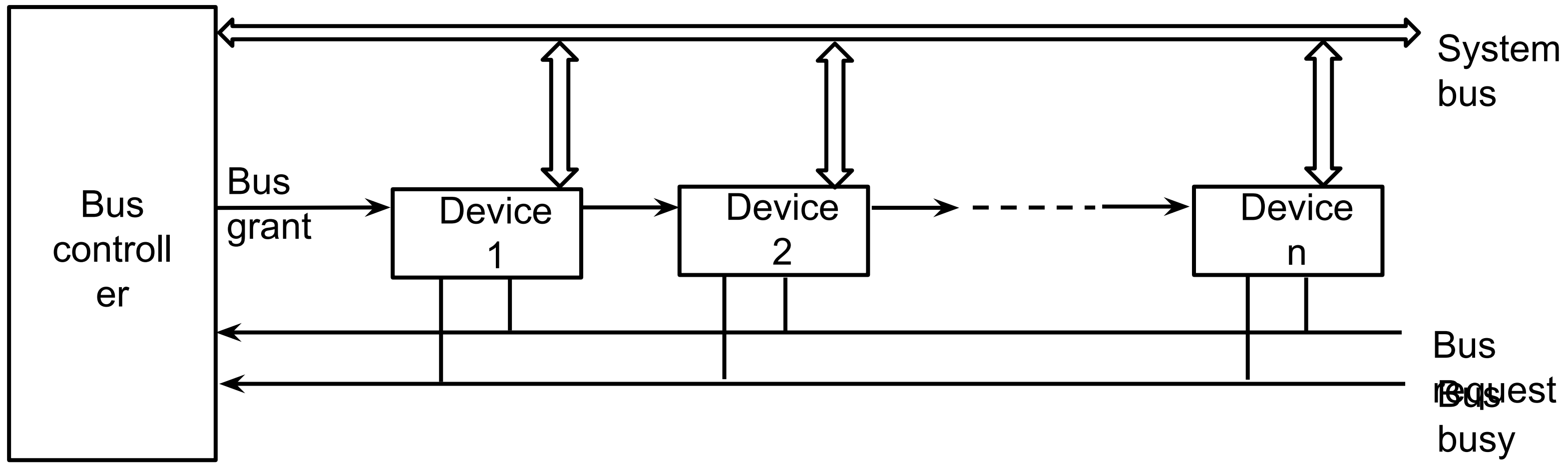
Bus Arbitration

- Bus arbitration means the decision of which device gets the access of buses for reading and writing operation.
- When multiple devices may need to use the buses at the same time then bus conflict will occur.
- To remove this bus conflict or bus contention bus arbitration is necessary.

There are three bus arbitration methods.

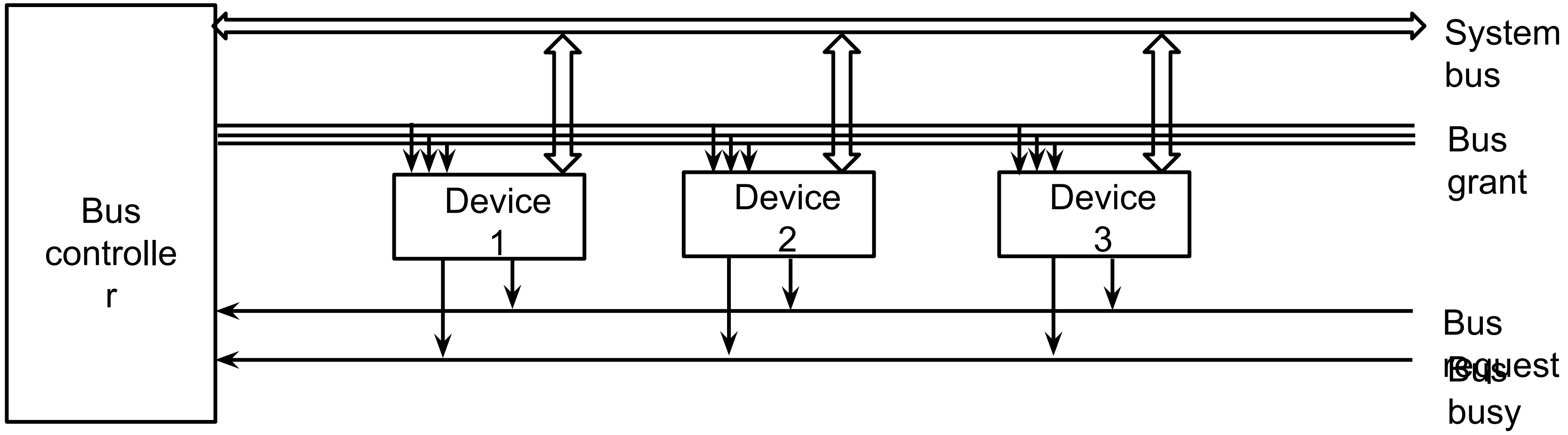
1. Daisy Chaining method
2. Polling method
3. Independent Request method

Daisy Chaining method



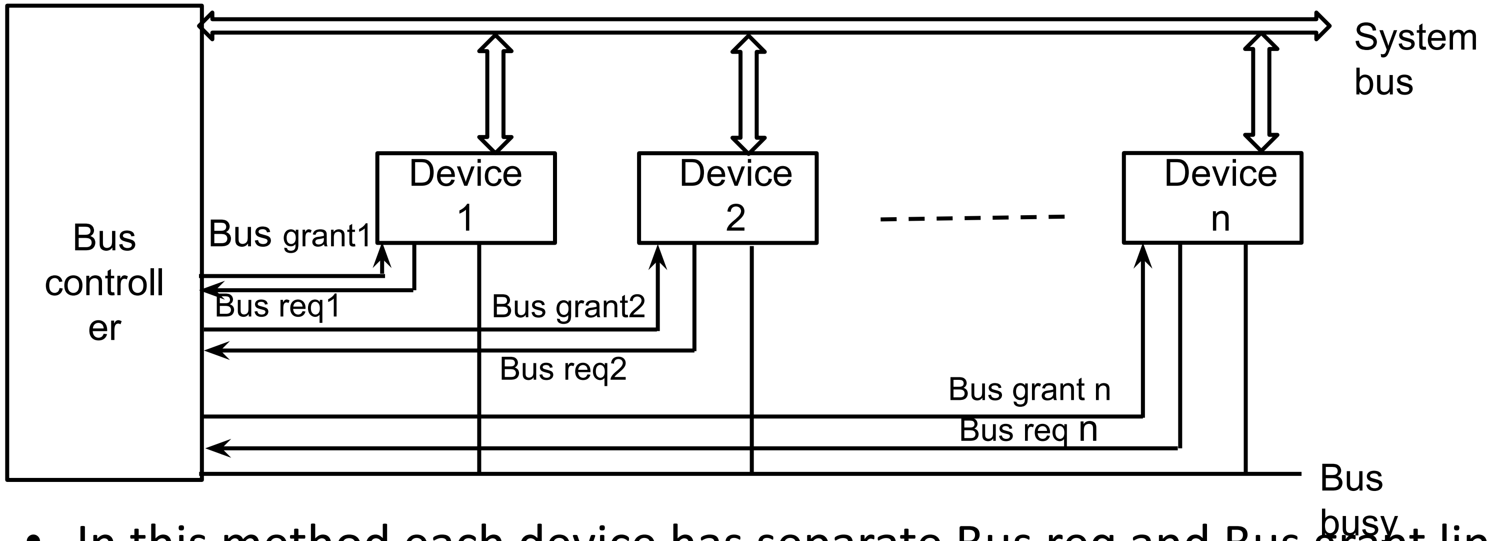
- In daisy chaining method every devices send their request to Bus controller using Bus request line.
- Which devices are the nearer to the bus controller their priority is higher.
- Here Device1 has a highest priority and Device n has a lowest priority.
- Bus controller check the bus busy signal if it is low (that means no other devices are using bus) then it generates the Bus grant signal.
- If there is a bus requirement of Device1 then it will use the buses and Bus busy signal goes to high otherwise it will forward the Bus grant signal to next device.
- The advantage of daisy chaining method is easy to implement.
- The disadvantage is the lower priority Devices have to wait for a longest period of time.

Polling method



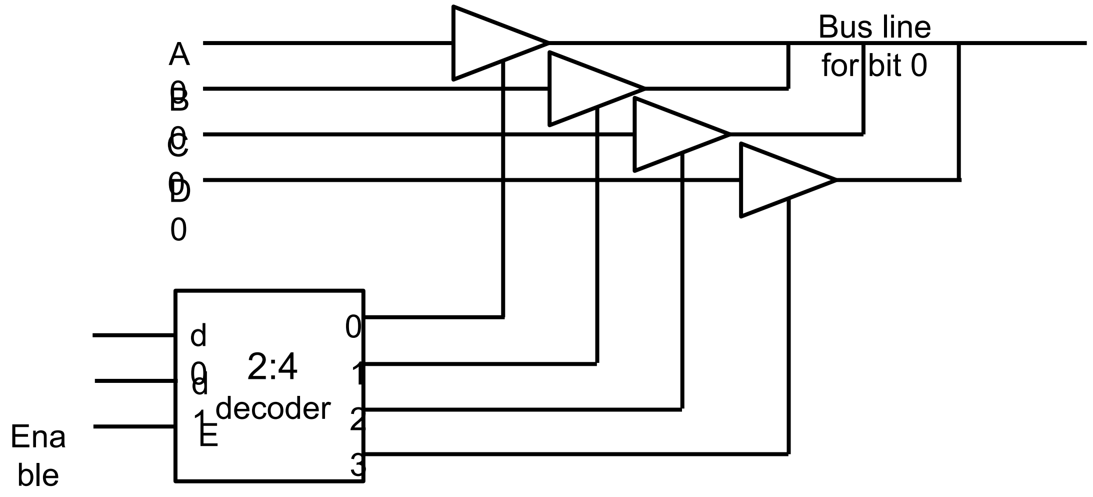
- Every devices can send their request to Bus controller using Bus request line.
- If the bus busy line is low then Bus controller checks the priority of devices and send the bus grant signal to this particular device.
- Here in this diagram three lines are used for Bus grant.
- Using three lines at most eight devices can connect.
- Take an example, for device1 Bus grant signal carry 000, for device2 001 and so on.
- Then bus busy signal goes to high.
- The advantage of this method is every device can get the Bus grant signal from bus controller directly.
- The disadvantage is extra poll lines are required for bus grant signal.

Independent Request method



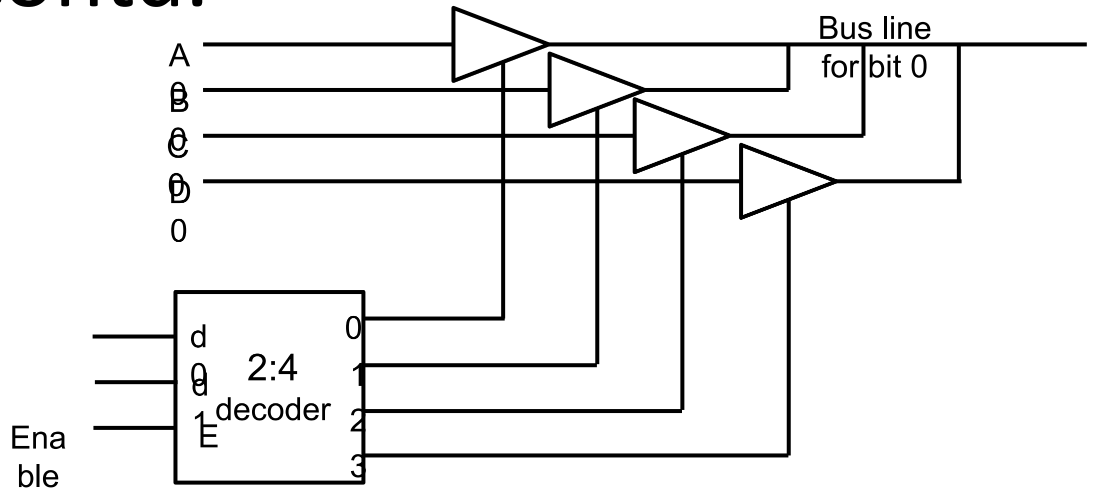
- In this method each device has separate Bus req and Bus grant line.
- Bus controller check the bus busy line if it is low then it generates the Bus grant signal to the particular device according to priority.
- Then Bus busy line goes to high.
- The advantage of daisy this method is each device can make an independent request and bus grant lines are also separated.
- The disadvantage is much more hardware required compared to other methods.

Bus organization using tri-state buffer



- Tri-state buffer is a digital circuit that exhibits three states. The states are logic 1 ,logic 0 and the third state is a high-impedance state (z).
- The high-impedance state behaves like an open circuit, which means that the output is disconnected and does not have logic significance.
- It has a normal input and a control input.
- The control input determines the output state. When the control input is equal to 1, the output is enabled and the gate behaves like any conventional buffer, with the output equal to the normal input.
- When the control input is 0, the output is disabled and the gate goes to a high-impedance state(Z).
- The outputs of four buffers are connected together to form a single bus line.

Bus organization using tri-state buffer contd.



- The control inputs to the buffers determine which of the four normal inputs will communicate with the bus line.
- No more than one buffer may be in the active state at any given time. The connected buffers must be controlled so that only one tri-state buffer has access to the bus line while all other buffers are maintained in a high- impedance state.
- One way to ensure that only one control input is active at any given time is to use a decoder, as shown in the diagram.

Thank You